LISTING OF THE CLAIMS

CLAIMS

We claim:

1. (Currently amended) A computerized method comprising: executing a nested transaction for computing data in an execution environment supporting a flat transaction only, and wherein a nested transaction encapsulates between a first StartTransaction operation and a corresponding first EndTransaction operation on a first nesting level a hierarchy of one or more further StartTransaction operations and corresponding further EndTransaction operations on further nesting levels, wherein a StartTransaction operation starts a transaction; and wherein an EndTransaction operation ends a transaction; wherein a facade library provides access from an object oriented environment to a relational database system, having a transaction object comprises a depth counter, a CommitTransaction operation, and a RollbackTransaction operation as object methods, and said method further comprising performing a StartTransaction operation by,

checking whether said StartTransaction operation is on the first testing level of said nested transaction, and

issuing a corresponding StartTransaction operation within said execution environment only ~~if~~ when said StartTransaction operation is on the first testing level of said nested transaction.

2. (Currently amended) A computerized method for executing a nested transaction for computing data in an execution environment supporting a flat transaction only according to claim 1, said method performing an EndTransaction operation by:

checking, in case said EndTransaction operation is a CommitTransaction operation successfully terminating a transaction, whether said EndTransaction operation is on said first nesting level of said nested transaction, and

issuing a corresponding CommitTransaction operation within said execution environment only ~~if~~ when said EndTransaction operation is on said first nesting level of said nested transaction.

3. (Previously presented) A computerized method for executing a nested transaction for computing data in an execution environment supporting a flat transaction only according to claim 2, said method performing an EndTransaction operation in case said EndTransaction operation is a RollbackTransaction operation aborting a transaction as unsuccessful, by issuing a corresponding RollbackTransaction operation within said execution environment independent from the nesting level of said RollbackTransaction operation.

4. (Previously presented) A computerized method for executing a nested transaction for computing data in an execution environment supporting a flat transaction only according to claim 3, said method performing, once a RollbackTransaction operation has been executed within said nested transaction, any further StartTransaction operation or any further EndTransaction operation within said nested transaction independent from its nesting level by rejecting it as being in error without issuing a corresponding StartTransaction operation or a corresponding EndTransaction operation to the execution environment.

5. (Currently amended) A computerized method for executing a nested transaction for computing data in an execution environment supporting a flat transaction only, and wherein a nested transaction encapsulates between a first StartTransaction operation and a corresponding first EndTransaction operation on a first nesting level a hierarchy of one or more further StartTransaction operations and corresponding further EndTransaction operations on further nesting levels, wherein a StartTransaction operation starts a transaction; and wherein an EndTransaction operation ends a transaction; and said method further comprising performing a StartTransaction operation by,

checking whether said StartTransaction operation is on the first testing level of said nested transaction, and

issuing a corresponding StartTransaction operation within said execution environment only ~~if~~ when said StartTransaction operation is on the first testing level of said nested transaction. said method performing an EndTransaction

operation by:

checking, in case said EndTransaction operation is a CommitTransaction operation successfully terminating a transaction, whether said EndTransaction operation is on said first nesting level of said nested transaction, and

issuing a corresponding CommitTransaction operation within said execution environment only ~~if~~ when said EndTransaction operation is on said first nesting level of said nested transaction;

said method performing an EndTransaction operation in case said EndTransaction operation is a RollbackTransaction operation aborting a transaction as unsuccessful, by issuing a corresponding RollbackTransaction operation within said execution environment independent from the nesting level of said RollbackTransaction operation;

said method performing, once a RollbackTransaction operation has been executed within said nested transaction, any further StartTransaction operation or any further EndTransaction operation within said nested transaction independent from its nesting level by rejecting it as being in error without issuing a corresponding StartTransaction operation or a corresponding EndTransaction operation to the execution environment, and

wherein said method:

checks the nesting level of any of said StartTransaction or EndTransaction operations by a depth counter,

increments said depth counter in the case of processing a StartTransaction operation, and

decrements said depth counter in the case of processing an EndTransaction operation which is a CommitTransaction operation, and

sets said depth counter to zero or an invalid value in the case of processing an EndTransaction operation which is a RollbackTransaction operation.

6. (Previously presented) A computerized method for executing a nested transaction for computing data in an execution environment supporting a flat transaction only according to claim 5, wherein:

said method is performed by a facade library separate from said execution environment, and said execution environment is a database system, and

said facade library provides access from an object oriented environment to said relational database system.

7. (Previously presented) A computerized method for executing a nested transaction for computing data in an execution environment supporting a flat transaction only according to claim 6, wherein said facade library comprises a STORE object class providing access to said database system and said STORE object class providing said StartTransaction operation as one of its methods.

8. (Previously presented) A computerized method for executing a nested transaction for computing data in an execution environment supporting a flat transaction only according to claim 7, said method performing said
StartTransaction operation by creating a transaction object for further control of said nested transaction in case said StartTransaction operation is on the first nesting level

9. (Previously presented) A computerized method for executing a nested transaction for computing data in an execution environment supporting a flat transaction only according to claim 8, wherein said transaction object comprises said depth counter, said CommitTransaction operation, and said RollbackTransaction operation as object methods.

10. (Currently amended) A computer implemented system for executing a nested transaction for computing data in an execution environment supporting a flat transaction only, the nested transaction being tangibly embodied in a computer to perform execution of transactions, and wherein a nested transaction encapsulates between a first StartTransaction operation and a corresponding first EndTransaction operation on a first nesting level a hierarchy of one or more further StartTransaction operations and corresponding further EndTransaction operations ends a transaction; said system performing a StartTransaction operation, the system comprising.

means for checking whether said StartTransaction operation is on the first nesting level of said nested transaction, and

means for issuing a corresponding StartTransaction operation within said execution environment only ~~if~~ when said StartTransaction operation is on the first testing level of said nested transaction.

11. (Currently amended) A data processing program for execution in a data processing system comprising software code portions for performing a computerized method for executing a nested transaction for computing data in an execution environment supporting a flat transaction only, and wherein a nested transaction encapsulates between a first StartTransaction operation and a corresponding first EndTransaction operation on a first nesting level a hierarchy of one or more further StartTransaction operations and corresponding further EndTransaction operations on further nesting levels, wherein a StartTransaction operation starts a transaction; and wherein an EndTransaction operation ends a transaction; wherein a facade library provides access from an object oriented environment to a relational database system, having a transaction object comprises a depth counter, a CommitTransaction operation, and a RollbackTransaction operation as object methods, and said method further comprising performing a StartTransaction operation by,

checking whether said StartTransaction operation is on the first nesting level of said nested transaction, and

issuing a corresponding StartTransaction operation within said execution environment only ~~if~~ when said StartTransaction operation is on the first nesting level of said nested transaction.

12. (Currently amended) A computer program product stored on a computer usable medium, comprising computer readable program means for causing a computer to perform a method executing a nested transaction in an execution environment supporting a flat transaction only, and wherein a nested transaction encapsulates between a first StartTransaction operation and a corresponding first EndTransaction operation on a first nesting level a hierarchy of one or more further StartTransaction operations and corresponding further EndTransaction operations on further nesting levels, wherein a StartTransaction operation starts a transaction; and wherein an EndTransaction operation ends a transaction; wherein a facade library provides access from an

object oriented environment to a relational database system, having a transaction object comprises a depth counter, a CommitTransaction operation, and a RollbackTransaction operation as object methods, and said method further comprising performing a StartTransaction operation by,

checking whether said StartTransaction operation is on the first nesting level of said nested transaction,

and issuing a corresponding StartTransaction operation within said execution environment only ~~if~~ when said StartTransaction operation is on the first testing level of said nested transaction.

13. (Previously presented) An article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for causing execution of a nested transaction for computing data in an execution environment supporting a flat transaction only, the computer readable program code means in said article of manufacture comprising computer readable program code means for causing a computer to effect the steps of claim 1.

14. (previously presented) A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for, said method steps comprising the steps of claim 1.

15. (Previously presented) An article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for causing execution of a nested transaction for computing data in an execution environment supporting a flat transaction only, the computer readable program code means in said article of manufacture comprising computer readable program code means for causing a computer to effect the steps of claim 5.

16. (previously presented) A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for, said method steps comprising the steps of claim 5.

17. (Previously presented) A computer program product comprising a computer usable medium having computer readable program code means embodied therein for causing a executing a nested

transaction for computing data in an execution environment supporting a flat transaction only, the computer readable program code means in said computer program product comprising computer readable program code means for causing a computer to effect the functions of claim 10.

18. (Currently amended) A computerized method for executing a nested transaction in an execution environment supporting a flat transaction only according to claim 1, said method performing an EndTransaction operation for computing data by:

checking, in case said EndTransaction operation is a CommitTransaction operation successfully terminating a transaction, whether said EndTransaction operation is on said first nesting level of said nested transaction, and

issuing a corresponding CommitTransaction operation within said execution environment only ~~if~~ when said EndTransaction operation is on said first nesting level of said nested transaction;

said method performing an EndTransaction operation in case said EndTransaction operation is a RollbackTransaction operation aborting a transaction as unsuccessful, by issuing a corresponding RollbackTransaction operation within said execution environment independent from the nesting level of said RollbackTransaction operation; and

said method performing, once a RollbackTransaction operation has been executed within said nested transaction, any further StartTransaction operation or any further EndTransaction operation within said nested transaction independent from its nesting level by rejecting it as being in error without issuing a corresponding StartTransaction operation or a corresponding EndTransaction operation to the execution environment.

19. (Previously presented) An article of manufacture comprising a computer usable medium having computer readable program code means embodied therein for causing execution of a nested transaction for computing data in an execution environment supporting a flat transaction only, the computer readable program code means in said article of manufacture comprising computer readable program code means for causing a computer to effect the steps of claim 18.

20. (Previously presented) A computerized method for executing a nested transaction in an execution environment supporting a flat transaction only according to claim 5, wherein:

said method is performed by a facade library separate from said execution environment, and said execution environment is a database system, and

said facade library provides access from an object oriented environment to said relational database system;

said facade library comprises a STORE object class providing access to said database system and said STORE object class providing said StartTransaction operation as one of its methods;

said method performing said StartTransaction operation by creating a transaction object for further control of said nested transaction in case said StartTransaction operation is on the first nesting level; and

said transaction object comprises said depth counter, said CommitTransaction operation, and said RollbackTransaction operation as object methods.

REMARKS

These remarks follow the order of the paragraphs of the office action. Relevant portions of the office action are shown indented and italicized.

DETAILED ACTION

1. This Office Action is in response to the applicants' communication received on August 18, 2006.

2. Claims 1-20 are presented for examination.

3. The applicants have amended claims 1-13, 15, and 17 and have added new claims 18-20 in the amendment received on August 18, 2006.

4. Applicants' arguments with respect to claims 1-20 have been considered but are deemed to be moot in view of the new grounds of rejection.

Claim Rejections -36 USC § 112

6. The following is a quotation of the second paragraph of 35 U.S.C. 112: The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

7. Claims 1,5, and 10-12 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.
Claims 1, 5, and 10-12 comprise an algorithm that begins with a Start Transaction and an IF conditional statement. These statements are conditional statements within an algorithm. These statements are a part of a series of looping statements. These looping statements do not contain an end to the conditional statements that would enable the statements to produce a positive result. It is the examiner's request to the applicants] that the applicants should kindly consider placing a statement after these conditional statements, in an effort to have the algorithm fall out of the loop with a statement that contains positive results.

In response, the applicants respectfully express their appreciation for the suggestion. Claims 1,5, and 10-12 are amended herein to remove the if condition and to substitute a statement that ends by replacing the word 'if' with the word 'when'. This overcomes the rejection of Claims 1,5, and 10-12 under 35 U.S.C. 112, second paragraph. Claims 1,5, and 10-12 are definite and

particularly point out and distinctly claim the subject matter which applicants regard as the invention. Thus Claims 1,5, and 10-12 are allowable.

Applicants note that Claims 1,5, and 10-12 2, and 18 are similarly amended herein.

Claim Rejections - 35 USC § 102

8. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:
A person shall be entitled to a patent unless -
(e) the invention was described in (1) an application for patent, published under section 122(b), by another lied In the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another flied in the United States before the Invention by the applicant for patent, except that an international application lied under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

9. Claims 1-20 are rejected under 35 U.S.C. 102(e) as being anticipated by Robertson et al. (Publication No.: US 200510240621 Al filed June 24, 2005, priority to provisional application no. 60/206564 filed on May 22, 2000, hereinafter Robertson).

In response, the applicant respectfully states that Claims 1-20 are apparently not anticipated by the invention of Robertson. The present invention, claimed in Claims 1-20, discloses:

"a means and a method for executing a nested transaction in an execution environment supporting flat transactions only. To process a StartTransaction operation within a nested transaction it is suggested to check whether the StartTransaction operation is on the first nesting level of the nested transactions. An actual transaction within the execution environment by issuing a corresponding StartTransaction is started only in the affirmative case but not otherwise. To process a CommitTransaction operation within a nested transaction to successfully terminate a transaction it is suggested to check whether the CommitTransaction operation is on the first nesting level of the nested transaction An actual transaction within the execution environment will be terminated only by issuing a corresponding CommitTransaction operation in the affirmative case but not otherwise To process a RollbackTransaction operation within a nested transaction aborting a transaction

as unsuccessful it is suggested to issue a corresponding RollbackTransaction operation within the execution environment independent from the nesting level of said RollbackTransaction operation.

Whereas, the cited art to Robertson, US Patent Publication No.: US 2005/0240621, filed: June 24, 2005, is entitled: "Method and system for managing partitioned data resources". The Robertson abstract reads:

"In accordance with an exemplary embodiment of the present invention, association forming entities are: a) maintained as objects in a like manner to the data objects being associated; and 6) are themselves partitioned objects comprising two or more association fragments, each association fragment being mostly concerned with the interfaces to a particular data object participating in the association. In accordance with an exemplary embodiment of the present invention, each association fragment affiliated with a particular data object is stored in a location that enhances the ease of interaction between the association fragment and the data object. For example, where a first data object and second data object are maintained in data stores at some distance from one another, physically or logically, then a first association fragment will be located with or near to the first data object and a second association fragment will be located with or near the second data object, at least within the same partition. This arrangement may be preferable because the volume of interaction between a data object and its respective association fragment may far outweigh the interaction needed between the two association fragments. This arrangement may also be preferable as the volume of interaction between a client application and both the data object and respective association fragment may exceed the interaction needed between the two association fragments. Some interactions will employ only one of the association fragments with the net result being a reduction in communications requirements and an improvement in performance. The present invention further provides for defining logical domains which are arbitrary and entirely orthogonal to partitions."

Thus as recognized by the office communication Robertson is concerned with managing partitioned data resources. Robertson is apparently not concerned with executing a nested

transaction in an execution environment supporting flat transactions, as in claims 1-20. Thus claims 1-20 are not anticipated by Robertson and are allowable.

> *10. Regarding Claims 1, 5, and 1 0-12, Robertson teaches a method and system for managing partitioned data resources.*
> *The method and associated system for managing partitioned data resources as taught or suggested by Robertson includes: nested transaction encapsulates between a StartTransaction operation (paragraphs 0275-0277) and a corresponding EndTransaction operation (paragraphs 0275-0277) on a hierarchy or further StartTransaction operations corresponding to further EndTransaction operations on nesting levels (paragraphs 0275-0277), a façade library (paragraph 0294) provides access from an object oriented environment to a relational database system, having a transaction object comprises a depth counter (paragraph 0325), a CommitTransaction operation (paragraph 0232), end a Rollback Transaction operation as object methods (paragraph 0204)! StartTransaction on the first level (paragraphs 0275-0277) and issuing a corresponding operation only if StartTransaction operation is on the first revel of the nested transaction (paragraphs 0275-0277).*

In response, the applicants respectfully state that a review of the cited portions of Robertson fails to show that Robertson teaches the method or system claimed in Claims 1, 5, and 1 0-12. For example, claim 1 as amended reads:

> 1. A computerized method comprising: executing a nested transaction for computing data in an execution environment supporting a flat transaction only, and wherein a nested transaction encapsulates between a first StartTransaction operation and a corresponding first EndTransaction operation on a first nesting level a hierarchy of one or more further StartTransaction operations and corresponding further EndTransaction operations on further nesting levels, wherein a StartTransaction operation starts a transaction; and wherein an EndTransaction operation ends a transaction; wherein a facade library provides access from an object oriented environment to a relational database system, having a transaction object comprises a depth counter, a CommitTransaction operation, and a RollbackTransaction operation as object methods, and said method further comprising performing a StartTransaction operation by,
>
> checking whether said StartTransaction operation is on the first testing level of said nested transaction, and

issuing a corresponding StartTransaction operation within said execution environment only ~~if~~ when said StartTransaction operation is on the first testing level of said nested transaction.

Robertson is not concerned with and doesn't allude to a flat transaction to which the present claims are directed. Furthermore, applicants take exception with the office communication statement above that,

> *Robertson includes: nested transaction encapsulates between a StartTransaction operation (paragraphs 0275-0277) and a corresponding EndTransaction operation (paragraphs 0275-0277) on a hierarchy or further StartTransaction operations corresponding to further EndTransaction operations on nesting levels (paragraphs 0275-0277),* etc.

which allegedly indicates that Robertson anticipates Claims 1, 5, and 1 0-12. A review of the cited portions of Robertson shows that Robertson does not perform or allude to the steps of executing, checking or issuing of claim 1. For example, Robertson [0275] - [0277] reads:

> [0275] With regard to FIGS. 13A-13B, a flowchart depicting the transaction process employed by the transaction manager is illustrated in accordance with a preferred embodiment of the present invention. The process begins with the client sending a transaction request to the transaction manager (step 1302). The transaction manager may be any of transaction managers 912A1-912AN depicted in FIG. 9 and the client may be one of the services being run in containers 906. Upon receiving the request, the transaction manager creates a transaction for the client, issues an enterprise lease for the transaction, and then returns the transaction content to the client as a transaction object (TXN) (step 1304). The transaction manager will manage the transaction only as long as a valid enterprise lease exists for the transaction. Should the enterprise lease expire, the transaction manager will clean up the client's transaction. By using the enterprise leasing concept, the client need not notify the transaction manager in case of a transaction failure. Should the client not be able to complete a transaction, the transaction manager automatically cleans up after the enterprise lease expires. However, if the transaction is proceeding at a slower than expected pace, the client can always renew the enterprise lease with the transaction manager. The enterprise lease maintenance process will not be further described for the transaction manager as the process has been fully described for the registrar with respect to FIG. 11B.
>
> [0276] Returning to FIG. 13A, the client then requests various resources to join the transaction by passing the TXN to a resource in a request to join the transaction (step 1306). Recall that the client may be communicating with resources via traditional resource managers or might instead be communicating with NW-enable devices and network elements that are managing a resource. Next, if the resource manager notifies the client that it intends to join the transaction, the process flows to step 1308 where the resource

> manager also passes a participant interface to the transaction manager (step 1314). The client then determines whether or not another resource is needed in the transaction (step 1316). If another resource is necessary, the process returns to step 1306 where the client invites another resource to join the transaction and the process continues as described immediately above. If, on the other hand, another resource is not necessary for the transaction and the client has joined the necessary resources in the transaction, the client makes a "commit" call to the transaction manager (step 1318). At that point, the transaction manger implements the two-phase commit process which is invoked by the transaction manager on all participants joined in the current transaction (step 1318).
>
> [0277] Returning to step 1308, should the resource manager not join the transaction, either expressly or by failing to respond to the client's request, the client may attempt to find another resource (step 1312). The process implemented by the client for finding a resource is similar to that described above with respect to FIG. 10 for finding a service and will be discussed further with respect to the DataBus. Should the client find another resource that is suitable for the transaction, the process reverts to step 1306 and continue as described above. However, if the client cannot find a suitable resource to transact with, the transaction ends. At some point the transaction's enterprise lease expires with the transaction manager and the transaction manager cleans up itself

Although the word 'transaction' occurs in Robertson [0275] - [0277], there is no reference or allusion in this portion of Robertson to 'a nested transaction', no reference or allusion in this portion of Robertson to 'a flat transaction', no reference or allusion in this portion of Robertson to ' a nested transaction that is encapsulated between a first StartTransaction operation and a corresponding first EndTransaction operation, no reference or allusion in this portion of Robertson to 'a first nesting level a hierarchy'.

Furthermore, applicants take exception with the office communication statement above that, the referenced portion of Robertson corresponds or anticipates the claimed invention by alleging:

> a façade library (paragraph 0294) provides access from an object oriented environment to a relational database system, having a transaction object comprises a depth counter (paragraph 0325), a CommitTransaction operation (paragraph 0232), end a Rollback Transaction operation as object methods (paragraph 0204)! StartTransaction on the first level (paragraphs 0275-0277) and issuing a corresponding operation only if StartTransaction operation is on the first revel of the nested transaction (paragraphs 0275-0277).

Robertson [294] reads:

[0294] The DataBus comprises a data layer with an object-oriented interface. All interactions with data are through methods on objects. If relational technology is used for actual backing store, then this implies that the data layer is actually two distinct tiers: 1) a persistent business object layer; and 2) the underlying relational database which stores the state of these business objects. If object-oriented database technology is used, the data layer might be realized as a single tier. In any case, we assume throughout this document that a distributed object-oriented approach is applied to the entire architecture, even if the wording of this document sometimes lapses into database terminology.

There apparently is no reference or allusion in this portion of Robertson to 'a facade library that provides access from an object oriented environment to a relational database system'.

The other cited portion of Robertson [325] reads:

[0325] One of the main roles of this central manager is to provide coordination and management of unique primary keys (PKs) across all partitions. In the present architecture, all entities follow the convention of defining a candidate primary key consisting of a unique 64-bit integer called the UID (unique identifier). This UID provides a convenient foreign key that is used by externalized association engines to store references to entity instances, as will be further described below. In accordance with one exemplary embodiment of the present invention, one of the primary responsibilities of central entity manager 2010 is to maintain a block-up counter for generating new UIDs when a new block of primary keys is called for by any of satellites 2012-2018. Satellites 2012-2018 actually issue a primary key whenever an entity instance is created and not steward 2010. This approach avoids the necessity of accessing the manager upon every creation of a new entity instance. The satellite only need consult the steward during entity creation in the event that the satellite runs out of keys in its allocated block of keys. It must then go back to the steward to request another block of keys. This approach avoids the necessity of accessing the manager upon every creation of a new entity instance. The satellite only need consult the steward during entity creation in the even that the satellite runs out of keys In accordance with another exemplary embodiment of the present

> invention, steward 2010 validates that a primary key proposed by a user for a new instance is not already in use by an existing instance. This latter sort of PK contrasts with the block-up UID generated by the central manager in that its form is dictated by the type of business object it represents. For example, the PK for a given entity might be a string or an integer, or it might be a composite key having more than one component. These domain-specific PKs would often be proposed by the application, or by custom logic within the entity implementation, and checked for uniqueness by the central entity manager, using for example, a hashing or directory service

There is no reference or allusion in this portion of Robertson to 'a transaction object which comprises a depth counter'.

The other cited portions of Robertson [232] and [204] read:

> [0232] Because of the use of Jini transactions, resources used in the GIB that are expected to participate in transactions must be managed by resource managers exposing a 2-phase commit interface, such as XA or the Jini transaction participant interface. A typical resource manager is a database management system, for example, Oracle or Versant. However, recall that in FIG. 8 enterprise network elements might be NW enabled and thus access NW services on GIB 802 by incorporating NW distributive intelligence services 824 onboard. In those cases, network elements 824 plug directly into GIB 802. In those cases, the network elements implement the relevant XA interfaces XAResource and XAConnection so that it could participate in the transaction as part of a transaction.
>
> [0204] This technique of enterprise leasing facilitates the implementation of self-healing services. If a process, on whose behalf a resource is leased, should abruptly crash, the lease will eventually expire and the system can de-allocate the resource. Things get cleaned up all by themselves. Moreover, with respect to the registrar, whenever an enterprise lease for service expires, the registrar can notify the self-healing services of the lease expiration. The self-healing services can then attempt to restart the service, either in the same or different container. In certain case, a process might include several transactions that are dependent on one another. If, as will be discussed below, a transaction has not been completed, the participants will be instructed by transaction manager 912 to roll back the process to a state prior to the commencement of the transaction, thus the participants are unaffected by a failure during a transaction. However, if several transactions have been successfully accomplished, the results of those transactions might be cached to a storage resource awaiting further processing. If the service hosting the resource fails, or even if a service fails that is crucial to the remaining

> transactions, it is likely that the cached data will not be recoverable. In the best case, the client can restart the process for the beginning and reestablish the data. In the worst case, the states of the service resources being used have been changed during the previous transactions making restarting the process impossible. The solution is a mirror resource that mirrors inter-process results for a running process. In case of a failure resulting in a lease expiration (service, communications or resource), the client or the client proxy maintains an object for the mirror and when the self-healing services restart the service(s), the client can continue the process with the mirrored interim results.

Although the cited portion mentions 'exposing a 2-phase commit interface', and "instructed by transaction manager 912 to roll back," there is indeed no reference or allusion in these portions of Robertson to 'a CommitTransaction operation', or 'a RollbackTransaction operation' as object methods.

In the previously cited (paragraphs 0275-0277) of Robertson, there is no reference or allusion to 'checking' an operation, 'a testing level', or a 'nested operation'. There is certainly no reference or allusion in this portion of Robertson to a step of 'checking whether a StartTransaction operation is on the first testing level of a nested transaction. There is no reference or allusion in this portion of Robertson to a step of 'issuing a corresponding StartTransaction operation within said execution environment' based on a test, when the StartTransaction operation is on the first testing level of a nested transaction. Thus although there are some words and even some phrases in Robertson that are the same as words in claim 1, these words and or phrases are used in different ways and in a different context than as used in the elements of claim 1. Thus Robertson indeed does not anticipate claim 1 which is allowable over the cited art. Also, all claims that depend on claim 1 are allowable over Robertson each for itself and because each depends on an allowable claim.

The remarks made regarding claim 1, are similarly applicable to Claims 5, and 10-12. Thus Claims 1, 5, and 1 0-12, and all claims that depend on any one of these are allowable over Robertson.

> *11. Regarding Clams 2-4, 8-9, and 13-20, the limitations of these claims have been noted in the rejection above. They are therefore rejected as set forth above.*

In response, the applicants respectfully state that the remarks made regarding claim 1, are similarly applicable to Clams 2-4, 8-9, and 13-20. Thus Clams 2-4, 8-9, and 13-20, are allowable over Robertson each for itself and because each depends on an allowable claim.

It is anticipated that this amendment shows that claims 1-20 are allowable. If any question remains, please contact the undersigned before issuing a communication with a FINAL status.

Please charge any fee necessary to enter this paper or any other paper to deposit account 50-0510.

Respectfully submitted,

By: _____/Louis Herzberg/_____
Dr. Louis P. Herzberg
Reg. No. 41,500
Voice Tel. (845) 352-3194
Fax. (845) 352-3194

3 Cloverdale Lane
Monsey, NY 10952

Customer Number: 54856